Approximating the optimal competitive ratio for an ancient online scheduling problem

Lin Chen[1] Deshi Ye[1] Guochuan Zhang[1]
[1]College of Computer Science, Zhejiang University, Hangzhou, 310027, China
chenlin198662@zju.edu.cn, zgc@zju.edu.cn

Abstract

We consider the classical online scheduling problem $P||C_{max}$ in which jobs are released over list and provide a nearly optimal online algorithm. More precisely, an online algorithm whose competitive ratio is at most $(1+\epsilon)$ times that of an optimal online algorithm could be achieved in polynomial time, where m, the number of machines, is a part of the input. It substantially improves upon the previous results by almost closing the gap between the currently best known lower bound of 1.88 [21] and the best known upper bound of 1.92 [15]. It has been known by folklore that an online problem could be viewed as a game between an adversary and the online player. Our approach extensively explores such a structure and builds up a completely new framework to show that, for the online over list scheduling problem, given any $\epsilon > 0$, there exists a uniform threshold K which is polynomial in m such that if the competitive ratio of an online algorithm is $\rho \le 2$, then there exists a list of at most K jobs to enforce the online algorithm to achieve a competitive ratio of at least $\rho - O(\epsilon)$. Our approach is substantially different from that of [19], in which an approximation scheme for online over time scheduling problems is given, where the number of machines is fixed. Our method could also be extended to several related online over list scheduling models.

Keywords: Competitive analysis; Online scheduling; Dynamic programming.

1 Introduction

Very recently Günther et al. [19] come up with a nice notion called *Competitive ratio approximation scheme* for online problems. Formally speaking, it is a series of online algorithms $\{A_\epsilon : \epsilon > 0\}$, where A_ϵ has a competitive ratio at most $(1+\epsilon)$ times the optimal competitive ratio. Naturally, a competitive ratio approximation scheme could be seen as an online version of the PTAS (polynomial time approximation scheme) for the offline problems. Using such a notion, they provide nearly optimal online algorithms for several online scheduling problems where jobs arrive over time, including $Qm|r_j,(pmtn)|\sum w_jc_j$ as well as $Pm|r_j|C_{max}$, where m is the number of machines. The algorithm runs in polynomial time when m is fixed.

That is a great idea for designing nearly optimal online algorithms, that motivates us to revisit the classical online problems which still have a gap between upper and lower bounds. However, the technique of Günther et al. [19] heavily relies on the structure of the optimal solution for the over time scheduling problem, through which they can focus on jobs released during a time window of a constant length. It thus seems hard to generalize to other online models.

Clearly, the first online scheduling problem which should be revisited is $P||C_{max}$, a fundamental problem in which jobs are released over list. This ancient scheduling model admits a simple

algorithm called LS (list scheduling) [18]. Its competitive ratio is $2 - 1/m$ that achieves the best possible for $m = 2, 3$ [14]. Nevertheless, better algorithms exist for $m = 4, 5, 6, 7$, see [10] [16] [22] for upper and lower bounds for online scheduling problems where m taking these specified values. Many more attentions are paid to the general case where m is arbitrary. There is a long list of improvements on upper and lower bounds, see [1] [7] [20] for improvements on competitive algorithms, and [1] [8] [17] for improvements on lower bounds. Among them the currently best known upper bound is $1 + \sqrt{\frac{1+\ln 2}{2}} \approx 1.9201$ [15], while the best known lower bound is 1.88 [21]. We refer the readers to [23] for a nice survey on this topic.

Although the gap between the upper and lower bounds are relatively small, it leaves a great challenge to close it. In this paper we tackle this classical problem by providing a competitive ratio approximation scheme. The running time is polynomial in the input size. More precisely, the time complexity related to m is $O(m^\Lambda)$ where $\Lambda = 2^{O(1/\epsilon^2 \log^2(1/\epsilon))}$. It is thus polynomial even when the number of machines is a part of the input.

To simplify the notion, throughout this paper we use *competitive scheme* instead of competitive ratio approximation scheme.

General Ideas We try to give a full picture of our techniques. Given any $\epsilon > 0$, at any time it is possible to choose a proper value (called a scaling factor) and scale all the jobs released so far such that there are only a constant number of different kinds of jobs. We then represent the jobs scheduled on each machine by a tuple (called a trimmed-state) in which the number of each kind of jobs remains unchanged. Composing the trimmed-states of all machines forms a trimmed-scenario and the number of different trimmed-scenarios we need to consider is a polynomial in m, subject to the scaling factors.

Given a trimmed-scenario, we can compute the corresponding approximation ratio (comparing with the optimal schedule), which is called an *instant approximation ratio*. Specifically, if the schedule arrives at a trimmed-scenario when the adversary stops, then the competitive ratio equals to the instant approximation ratio of this trimmed-scenario. Formal definitions will be given in the next section. Note that the instant approximation ratio of every trimmed-scenario could be determined (up to an error of $O(\epsilon)$) regardless of the scaling factor.

To understand our approach easily we consider the online scheduling problem as a game. Each time the adversary and the scheduler take a move, alternatively, i.e., the adversary releases a job and the online scheduler then assigns the job to a machine. It transfers the current trimmed-scenario into a new one. Suppose the adversary wins the game by leading it into a certain trimmed-scenario with an instant approximation ratio ρ, forcing the competitive ratio to be at least ρ. The key observation is that if he has a winning strategy, he would have a winning strategy of taking only a polynomial number (in m) of moves since the game itself consists of only a polynomial number of distinct trimmed-scenarios. A rigorous proof for such an observation relies on formulating the game into a layered graph and associating the scheduling of any online algorithm with a path in it. Given the observation, the online problem asks if the adversary has a winning strategy of $C = poly(m)$ moves, starting from a trimmed-scenario where there is no job. Such a problem could be solved via dynamic programming, which decomposes it into a series of subproblems that ask whether the adversary has a winning strategy of $C' < C$ moves, starting from an arbitrary trimmed-scenario.

Various extensions could be built upon this framework. Indeed, competitive schemes could be

achieved for $Rm||C_{max}$ and $Rm||\sum_i C_i^p$ where $p \geq 1$ is some constant and C_i is the completion time of machine i. The running times of these schemes are polynomial when m is a constant.

In addition to competitive schemes, it is interesting to ask if we can achieve an optimal online algorithm. We consider the semi-online model $P|p_j \leq q|C_{max}$, where all job processing times are bounded. We are able to design an optimal online algorithm running in $(mq)^{O(mq)}$ time. It is exponential in both m and q.

Recall that the competitive ratio of list scheduling for $P||C_{max}$ is $2 - 1/m$. Throughout the paper we focus on online algorithms whose competitive ratio is no more than 2. We assume that $m \geq 2$.

2 Structuring Instances

To tackle the online scheduling problem, similarly as the offline case we want to well structure the input instance subject to an arbitrarily small loss. However, in the online setting we are not aware of the whole input. The instance needs scaling in a dynamic way.

Given any $0 < \epsilon \leq 1/4$, we may assume that all the jobs released have a processing time of $(1+\epsilon)^j$ for some integer $j \geq 0$. Let c_0 be the smallest integer such that $(1+\epsilon)^{c_0} \geq 1/\epsilon$. Let ω be the smallest integer such that $(1+\epsilon)^{\omega} \geq 3$. Let $SC = \{(1+\epsilon)^{j\omega} | j \geq 0, j \in \mathbb{N}\}$.

Consider the schedule of n ($n \geq 1$) jobs by any online algorithm. Let $p_{max} = \max_j\{p_j\}$. Then $LB = \max\{\sum_{j=1}^n p_j/m, p_{max}\}$ is a trivial lower bound on the makespan. We choose $T_{LB} \in SC$ such that $T_{LB} \leq LB < T_{LB}(1+\epsilon)^{\omega}$, and define job j as a small job if $p_j \leq T_{LB}(1+\epsilon)^{-c_0}$, and a big job otherwise. T_{LB} is called the *scaling factor* of this schedule.

Let L_h^s be the load (total processing time) of small jobs on machine h. An $(\omega + c_0 + 1)$-tuple $st_h = (\eta^h_{-c_0}, \eta^h_{-c_0+1}, \cdots, \eta^h_{\omega})$ is used to represent the jobs scheduled on machine h, where η_i^h ($-c_0 + 1 \leq i \leq \omega$) is the number of big jobs with processing time $T_{LB}(1+\epsilon)^i$ on machine h, and $\eta^h_{-c_0} = L_s^h/(T_{LB}(1+\epsilon)^{-c_0})$. We call such a tuple as a state (of machine h). The first coordinate of a state might be fractional, while the other coordinates are integers. The load of a state is defined as $LD(st_h) = \sum_{i=-c_0}^{\omega}(1+\epsilon)^i\eta_i \leq 4LB$.

Composing the states of all machines forms a scenario $\psi = (st_1, st_2, \cdots, st_m)$. Thus, any schedule could be represented by (T_{LB}, ψ) where $T_{LB} \in SC$ is the scaling factor of the schedule. Specifically, if the adversary stops now, then the competitive ratio of such a schedule is approximately (up to an error of $O(\epsilon)$):

$$\rho(\psi) = C_{max}(\psi)/OPT(\psi)$$

where $C_{max}(\psi) = \max_j LD(st_j)$, and $OPT(\psi)$ is the makespan of an optimal solution for the offline scheduling problem in which jobs of ψ are taken as an input (here small jobs are allowed to split). We define $LD(\psi) = \sum_h LD(st_h)$ and $P_{max}(\psi)$ the largest processing time (divided by T_{LB}) of jobs in ψ ($P_{max}(\psi) = (1+\epsilon)^{-c_0}$ if there is no big job in ψ). Obviously,

$$OPT(\psi) \geq LB = \max\{LD(\psi)/m, P_{max}(\psi)\} \geq 1.$$

The above ratio is regardless of the scaling factor and is called an *instant approximation ratio.*

We can use a slightly different $(\omega + c_0 + 1)$-tuple $\tau = (\nu_{-c_0}, \nu_{-c_0+1}, \cdots, \nu_{\omega})$ to approximate a state, where each coordinate is an integer. It is called a *trimmed-state*. Specifically, τ is called a simulating-state of st_h if $\nu_i = \eta_i^h$ for $-c_0 < i \leq \omega$ and $\eta^h_{-c_0} \leq \nu_{-c_0} \leq \eta^h_{-c_0} + 2$.

We define $LD(\tau) = \sum_{i=-c_0}^{\omega} \nu_i(1+\epsilon)^i$ and restrict our attention on trimmed-states whose load is no more than $4LB + 2(1+\epsilon)^{-c_0}$. There are at most $\Lambda \le 2^{O(1/\epsilon^2 \log^2(1/\epsilon))}$ such kinds of trimmed-states (called feasible trimmed-states). We sort these trimmed-states arbitrarily as $\tau_1, \cdots, \tau_\Lambda$, and define a Λ-tuple $\phi = (\xi_1, \xi_2, \cdots, \xi_\Lambda)$ to approximate scenarios, where $\sum_i \xi_i = m$ and $0 \le \xi_i \le m$ is the number of machines whose corresponding trimmed-state is τ_i. Indeed, ϕ is called a trimmed-scenario and specifically, it is called a simulating-scenario of $\psi = (st_1, st_2, \cdots, st_m)$ if there is a one to one correspondence between the m states (i.e., st_1 to st_m) and the m trimmed-states of ϕ such that each trimmed-state is the simulating-state of its corresponding state.

Recall that in ψ, jobs are scaled with T_{LB}, thus $1 \le \max\{1/mLD(\psi), P_{max}(\psi)\} < (1+\epsilon)^\omega$. We may restrict our attentions to trimmed-scenarios satisfying $1 \le \max\{1/mLD(\phi), P_{max}(\phi)\} < (1+\epsilon)^\omega + 2(1+\epsilon)^{-c_0}$, where similarly we define $LD(\phi) = \sum_j \xi_j LD(\tau_j)$, and $P_{max}(\phi)$ the largest processing time of jobs in ϕ. Trimmed-scenarios satisfying the previous inequality are called feasible trimmed-scenarios.

Notice that there are $\Gamma \le (m+1)^\Lambda$ different kinds of feasible trimmed-scenarios. we sort them as $\phi_1, \cdots, \phi_\Gamma$. As an exception, we plug in two additional trimmed-scenarios ϕ_0 and $\phi_{\Gamma+1}$, where ϕ_0 represents the initial trimmed-scenario in which there are no jobs, and $\phi_{\Gamma+1}$ represents any infeasible trimmed-scenario. Let Φ be the set of these trimmed-scenarios. We define

$$\rho(\phi) = C_{max}(\phi)/OPT(\phi)$$

as the *instant approximation ratio* of a feasible trimmed-scenario ϕ, in which $C_{max}(\phi) = \max_j\{LD(\tau_j) : \xi_j > 0\}$, and $OPT(\phi)$ is the makespan of the optimum solution for the offline scheduling problem in which jobs of ϕ are taken as an input and every job (including small jobs) should be scheduled integrally. As an exception, we define $\rho(\phi_0) = 1$ and $\rho(\phi_{\Gamma+1}) = \infty$.

Furthermore, notice that except for $\phi_{\Gamma+1}$, $C_{max}(\phi) \le 4(1+\epsilon)^\omega + 2(1+\epsilon)^{-c_0} \le 20$, which is a constant. Thus we can divide the interval $[1, 20]$ equally into $19/\epsilon$ subintervals and let $\Delta = \{1, 1+\epsilon, \cdots, 1+\epsilon \cdot 19/\epsilon\}$. We round up the instant approximation ratio of each ϕ to its nearest value in Δ. For simplicity, we still denote the rounded value as $\rho(\phi)$.

Lemma 1 *If ϕ is a simulating-scenario of ψ, then $\rho(\psi) - O(\epsilon) \le \rho(\phi) \le \rho(\psi) + O(\epsilon)$.*

Proof. It can be easily seen that $OPT(\psi) \le OPT(\phi) \le OPT(\psi) + 3(1+\epsilon)^{-c_0}$. Meanwhile $C_{max}(\psi) \le C_{max}(\phi) \le C_{max}(\psi) + 2(1+\epsilon)^{-c_0}$. Note that $OPT(\psi) \ge 1$ and the lemma follows directly. □

Consider the scheduling of n jobs by any online algorithm. The whole procedure could be represented by a list as

$$(T_{LB}(1), \psi(1)) \to (T_{LB}(2), \psi(2)) \to \cdots \to (T_{LB}(n), \psi(n)),$$

where $\psi(k)$ is the scenario when there are k jobs, and $T_{LB}(k)$ is the corresponding scaling factor. Here $\psi(k)$ changes to $\psi(k+1)$ by adding a new job p_{k+1}, and the reader may refer to Appendix A to see how the coordinates of a scenario change when a new job is added.

Let μ_0 be the smallest integer such that $(1+\epsilon)^{\mu_0} \ge 4(1+\epsilon)^{\omega+c_0+1}$ and $R = \{0, (1+\epsilon)^{-c_0}, \cdots, (1+\epsilon)^{\lceil \mu_0/\omega \rceil + \omega - 1}\}$. We prove that, if a scenario ψ changes to ψ' by adding some job p_n, then there

exists some job $p'_n \in R$ such that ϕ changes to ϕ' by adding p'_n, and furthermore, ϕ and ϕ' are the simulating-scenarios of ψ and ψ', respectively. This suffices to approximate the above scenario sequence by the following sequence

$$\phi_0 \to \phi(1) \to \phi(2) \to \cdots \to \phi(n),$$

where $\phi(k)$ is the simulating-scenario of $\psi(k)$, and ϕ_0 is the initial scenario where there is no job.

We briefly argue why it is this case. Suppose T_{LB} is the scaling factor of ψ. According to the online algorithm, p_n is put on machine h where $st_h = (\eta_{-c_0}, \cdots, \eta_\omega)$. Let $\tau = (\nu_{-c_0}, \cdots, \nu_\omega)$ be its simulating state in ϕ. If $p_n/T_{LB} < (1+\epsilon)^{-c_0}$ and $\eta_{-c_0} + p_n/T_{LB} \leq \nu_{-c_0}$, then ϕ is still a simulating scenario of ψ' and we may set $p'_n = 0$. Else if $\nu_{-c_0} < \eta_{-c_0} + p_n/T_{LB} \leq \nu_{-c_0} + 1$, we may set $p'_n = (1+\epsilon)^{-c_0}$. For the upper bound on the processing time, suppose p_n/T_{LB} is so large that the previous load of each machine (which is no more than $4LB \leq 4(1+\epsilon)^\omega$) becomes no more than $(1+\epsilon)^{-c_0} p_n/T_{LB}$. It then makes no difference by releasing an even larger job. A rigorous proof involves a complete analysis of how the coordinates of a trimmed-scenario change by adding a job belonging to R (see Appendix B), and a case by case analysis of each possible changes between ψ and ψ' (see Appendix C).

3 Constructing a Transformation Graph

We construct a graph G that contains all the possible sequences of the form $\phi_0 \to \phi(1) \to \phi(2) \to \cdots \to \phi(n)$. This is called a transformation graph. For ease of our following analysis, some of the feasible trimmed-scenarios should be deleted. Recall that $1 \leq \max\{1/mLD(\phi), P_{max}(\phi)\} < (1+\epsilon)^\omega + 2(1+\epsilon)^{-c_0}$ is satisfied for any feasible trimmed-scenario ϕ, and it may happen that two trimmed-scenarios are essentially the same. Indeed, if $(1+\epsilon)^\omega \leq \max\{1/mLD(\phi), P_{max}(\phi)\} < (1+\epsilon)^\omega + 2(1+\epsilon)^{-c_0}$, then by dividing $(1+\epsilon)^\omega$ from the processing times of each job in ϕ we can derive another trimmed-scenario ϕ' satisfying $1 \leq \max\{1/mLD(\phi'), P_{max}(\phi')\} < 1 + 2(1+\epsilon)^{-c_0-\omega}$, which is also feasible. If ϕ is a simulating-scenario of ψ, then ϕ' is called a shifted simulating-scenario of ψ. It is easy to verify that the instant approximation ratio of a shifted simulating scenario is also similar to that of the corresponding scenario (see Appendix D). In this case ϕ is deleted and we only keep ϕ'. Let $\Phi' \subset \Phi$ be the set of remaining trimmed-scenarios. We can prove that, for any real schedule represented as $\psi(1) \to \psi(2) \to \cdots \to \psi(n)$, we can find $\phi_0 \to \phi(1) \to \phi(2) \to \cdots \to \phi(n)$ such that $\phi(k) \in \Phi'$ is either a simulating-scenario or a shifted simulating-scenario of $\psi(k)$. The reader can refer to Appendix D for a rigorous proof.

Recall that when a trimmed-scenario changes to another, the adversary only releases a job belonging to R. Let $\zeta = |R|$ and $\alpha_1, \cdots, \alpha_\zeta$ be all the distinct processing times in R. We show how G is constructed.

We first construct two disjoint vertex sets S_0 and A_0. For every $\phi_i \in \Phi'$, there is a vertex $s_i^0 \in S_0$. For each s_i^0, there are ζ vertices of A_0 incident to it, namely a_{ij}^0 for $1 \leq j \leq \zeta$. The node a_{ij}^0 represents the release of a job of processing time α_j to the trimmed-scenario ϕ_i. Thus, $S_0 \cup A_0$ along with the edges forms a bipartite graph.

Let $S_1 = \{s_i^1 | s_i^0 \in S_0\}$ be a copy of S_0. By scheduling a job of α_j, if ϕ_i could be changed to ϕ_k, then there is an edge between a_{ij}^0 and s_k^1. We go on to build up the graph by creating an arbitrary number of copies of S_0 and A_0, namely $S_1, S_2, \cdots$ and $A_1, A_2, \cdots$ such that $S_h = \{s_i^h | s_i^0 \in S_0\}$,

$A_h = \{a_{ij}^h | a_{ij}^0 \in A_0\}$. Furthermore, there is an edge between s_i^h and a_{ij}^h if and only if there is an edge between s_i^0 and a_{ij}^0, and an edge between a_{ij}^h and s_k^{h+1} if and only if there is an edge between a_{ij}^0 and s_k^1.

The infinite graph we construct above is the transformation graph G. We let G_n be the subgraph of G induced by the vertex set $(\cup_{i=0}^n S_i) \cup (\cup_{i=0}^{n-1} A_i)$.

4 Best Response Dynamics

Recall that We can view online scheduling as a game between the scheduler and the adversary. According to our previous analysis, we can focus on trimmed-scenarios and assume that the adversary always releases a job with processing time belonging to R. By scheduling a job released by the adversary, the current trimmed-scenario changes into another one.

We can consider the instant approximation ratio as the utility of the adversary who tries to maximize it by leading the scheduling into a (trimmed) scenario. After releasing n jobs, if he is satisfied with the current instant approximation ratio, then he stops and the game is called an n-stage game. Otherwise he goes on to release more jobs. The scheduler, however, tries to minimize the competitive ratio by leading the game into trimmed-scenarios with small instant approximation ratios.

Consider any n-stage game and define $\rho_n(s_k^n) = \rho(\phi_k)$. It implies that if the game arrives at ϕ_k eventually, then the utility of the adversary is $\rho(\phi_k)$. Notice that the adversary could release a job of processing time 0, thus n-stage games include k-stage games for $k < n$. Consider a_{ij}^{n-1}. If the current trimmed-scenario is ϕ_i and the adversary releases a job with processing time α_j, then all the possible schedules by adding this job to different machines could be represented by $N(a_{ij}^{n-1}) = \{s_k^n : s_k^n \text{ is incident to } a_{ij}^{n-1}\}$. The scheduler tries to minimize the competitive ratio, and he knows that it is the last job, thus he would choose the one with the least instant approximation ratio. Thus we define

$$\rho_n(a_{ij}^{n-1}) = \min_k \{\rho_n(s_k^n) : s_k^n \in N(a_{ij}^{n-1})\}.$$

Knowing this beforehand, the adversary chooses to release a job which maximizes $\rho_n(a_{ij}^{n-1})$. Let $N(s_i^{n-1}) = \{a_{ij}^{n-1} : a_{ij}^{n-1} \text{ is incident to } s_i^{n-1}\}$ and thus we define

$$\rho_n(s_i^{n-1}) = \max_j \{\rho_n(a_{ij}^{n-1}) : a_{ij}^{n-1} \in N(s_i^{n-1})\}.$$

Iteratively applying the above argument, we can define

$$\rho_n(a_{ij}^{h-1}) = \min_k \{\rho_n(s_k^n) : s_k^h \in N(a_{ij}^{h-1})\},$$

$$\rho_n(s_i^{h-1}) = \max_j \{\rho_n(a_{ij}^{h-1}) : a_{ij}^{h-1} \in N(s_i^{h-1})\}.$$

The value $\rho_n(s_i^h)$ means that, if the current trimmed-scenario is ϕ_i, then the largest utility the adversary could achieve by releasing $n - h$ jobs is $\rho_n(s_i^h)$. Notice that we start from the empty schedule s_0^0, thus $\rho_n(s_0^0)$ is the largest utility the adversary could achieve by releasing n jobs.

4.1 Bounding the number of stages

The computation of the utility of the adversary relies on the number of jobs released, however, theoretically the adversary could release as many jobs as he wants. In this section, we prove the following theorem.

Theorem 1 *There exists some integer $n_0 \le O((m+1)^\Lambda/\epsilon)$, such that $\rho_n(s_i^0) = \rho_{n_0}(s_i^0)$ for any $\phi_i \in \Phi'$ and $n \ge n_0$.*

To prove it, we start with the following simple lemmas.

Lemma 2 *For any $1 \le h \le n$, $\rho_n(s_i^h) \le \rho_n(s_i^{h-1})$.*

Proof. The proof is obvious by noticing that the adversary could release a job with processing time 0. □

Lemma 3 *For any $0 \le h \le n$ and $i \ne \Gamma + 1$, $\rho_n(s_i^h) \in \Delta$.*

Proof. The lemma clearly holds for $h = n$. Suppose the lemma holds for some $h \ge 1$, we prove that the lemma is also true for $h-1$.

Recall that $\rho_n(a_{ij}^{h-1}) = \min_k\{\rho_n(s_k^n) : s_k^h \in N(a_{ij}^{h-1})\}$. We prove that $\rho_n(a_{ij}^{h-1}) \in \Delta$. To this end, we only need to show that, we can always put α_j to a certain machine so that ϕ_i is not transformed into $\phi_{\Gamma+1}$.

We apply list scheduling when α_j is released. Suppose by scheduling α_j in this way, ϕ_i is transformed into $\phi_{\Gamma+1}$, then $\alpha_j = (1+\epsilon)^\mu$ for $1 \le \mu \le \omega$ and $LB' = \max\{1/mLD(\phi) + \alpha_j/m, P_{max}(\phi), \alpha_j\} < (1+\epsilon)^\omega + 2(1+\epsilon)^{-c_0}$. Furthermore, suppose α_j is put to a machine whose trimmed-state is τ. Then $LD(\tau) + \alpha_j \ge 4(1+\epsilon)^\omega + 2(1+\epsilon)^{-c_0}$. Now it follows directly that $LD(\tau) > 3(1+\epsilon)^\omega$. Notice that we put α_j to the machine with the least load. Before α_j is released, the load of every machine in ϕ_i is larger than $3(1+\epsilon)^\omega$, which contradicts the fact that ϕ_i is a feasible trimmed-scenario.

Therefore, applying list scheduling, ϕ_i can always transform to another feasible trimmed-scenario, which ensures that $\rho_n(a_{ij}^{h-1}) \in \Delta$. Thus $\rho_n(s_i^{h-1}) = \max_j\{\rho_n(a_{ij}^{h-1}) : a_{ij}^{h-1} \in N(s_i^{h-1})\} \in \Delta$.

□

Lemma 4 *If there exists a number $n \in N$ such that $\rho_{n+1}(s_i^0) = \rho_n(s_i^0)$, then for any integer $h \ge 0$, $\rho_{n+h}(s_i^0) = \rho_n(s_i^0)$.*

Proof. We prove the lemma by induction. Suppose it holds for h. We consider $h+1$.

Obviously $\rho_{n+h}(s_i^{n+h}) = \rho_{n+h+1}(s_i^{n+h+1}) = \rho(\phi_i)$. According to the computing rule,

$$\rho_{n+h+1}(a_{ij}^{n+h}) = \min_k\{\rho_{n+h+1}(s_k^{n+h+1}) : s_k^{n+h+1} \in N(a_{ij}^{n+h})\},$$

$$\rho_{n+h}(a_{ij}^{n+h-1}) = \min_k\{\rho_{n+h}(s_k^{n+h}) : s_k^{n+h} \in N(a_{ij}^{n+h-1})\}.$$

Recall that $s_k^{n+h+1} \in N(a_{ij}^{n+h})$ if and only if $s_k^1 \in N(a_{ij}^0)$, and thus it is also equivalent to $s_k^{n+h} \in N(a_{ij}^{n+h-1})$. Hence, $\rho_{n+h+1}(a_{ij}^{n+h}) = \rho_{n+h}(a_{ij}^{n+h-1})$.

Using analogous arguments, we can show that $\rho_{n+h+1}(s_i^{n+h}) = \rho_{n+h}(s_i^{n+h-1})$. Iteratively applying the above procedure, we can finally show that $\rho_{n+h+1}(s_i^1) = \rho_{n+h}(s_i^0)$. Similarly, $\rho_{n+h}(s_i^1) = \rho_{n+h-1}(s_i^0)$.

According to the induction hypothesis, we know $\rho_{n+h}(s_i^1) = \rho_{n+h-1}(s_i^0) = \rho_n(s_i^0)$, and $\rho_{n+h+1}(s_i^1) = \rho_{n+h}(s_i^0) = \rho_n(s_i^0)$. Meanwhile

$$\rho_{n+h}(a_{ij}^0) = \min_k\{\rho_{n+h}(s_k^1) : s_k^1 \in N(a_{ij}^0)\} = \min_k\{\rho_n(s_k^0) : s_k^1 \in N(a_{ij}^0)\},$$

$$\rho_{n+h+1}(a_{ij}^0) = \min_k\{\rho_{n+h+1}(s_k^1) : s_k^1 \in N(a_{ij}^0)\} = \min_k\{\rho_n(s_k^0) : s_k^1 \in N(a_{ij}^0)\}.$$

Thus it immediately follows that $\rho_{n+h}(a_{ij}^0) = \rho_{n+h+1}(a_{ij}^0)$. Furthermore,

$$\begin{aligned}
\rho_{n+h+1}(s_i^0) &= \max_j\{\rho_{n+h+1}(a_{ij}^0) : a_{ij}^0 \in N(s_i^0)\} \\
&= \max_j\{\rho_{n+h}(a_{ij}^0) : a_{ij}^0 \in N(s_i^0)\} = \rho_{n+h}(s_i^0).
\end{aligned}$$

The lemma holds for $h+1$. □

Now we arrive at the proof of Theorem 1. Define $Z(n) = \sum_{\phi_i \in \Phi' \setminus \{\phi_{\Gamma+1}\}} \rho_n(s_i^0)$ as the potential function. According to the previous lemmas, $Z(n+1) \geq Z(n)$, and if $Z(n_0+1) = Z(n_0)$, then $Z(n) = Z(n_0)$ for any $n \geq n_0$. Furthermore, if $Z(n+1) > Z(n)$, then $Z(n+1) - Z(n) \geq \epsilon$. Suppose $Z(n+1) > Z(n)$, then it follows directly that $Z(n+1) > Z(n) > \cdots > Z(1)$. Recall that $Z(1) \geq 0$ and $Z(n+1) \leq 20(|\Phi'|-1) \leq O((m+1)^\Lambda)$, thus $n+1 \leq O((m+1)^\Lambda/\epsilon)$. Furthermore, it can be easily verified that if $Z(n+1) = Z(n)$, then $\rho_{n+1}(s_i^0) = \rho_n(s_i^0)$ for any $\phi \in \Phi'$. Thus, by setting $n_0 = O((m+1)^\Lambda/\epsilon)$, Theorem 1 follows.

Let n_0 be the smallest integer satisfying Theorem 1. Let $\rho^* = \rho_{n_0}(s_0^0)$, and $\rho(s_i^0) = \rho_{n_0}(s_i^0)$. Now it is not difficult to see that, the optimal online algorithm for $P||C_{max}$ has a competitive ratio around ρ^*. A rigorous proof of such an observation depends on the following two facts.

1. Given any online algorithm, there exists a list of at most n_0 jobs such that by scheduling them, its competitive ratio exceeds $\rho^* - O(\epsilon)$.
2. There exists an online algorithm whose competitive ratio is at most $\rho^* + O(\epsilon)$.

The first fact could be proved via G_{n_0}, where $\rho^* = \rho_{n_0}(s_0^0)$ ensures that n_0 jobs are enough to achieve the lower bound. The readers may refer to Appendix E.1 for details. The second observation could be proved via G_{n_0+1}, where $\rho_{n_0+1}(s_i^0) = \rho_{n_0+1}(s_i^1) = \rho(s_i^0)$ for every ϕ_i. Each time a job is released, the scheduler may assume that he is at the vertex s_i^0 where $\rho_{n_0+1}(s_i^0) \leq \rho^*$, and find a feasible schedule by leading the game into s_k^1 where $\rho_{n_0+1}(s_i^0) = \rho_{n_0+1}(s_k^1) \leq \rho^*$. After scheduling the job he may still assume that he is at s_k^0. The readers may refer to Appendix E.2 for details.

Using the framework we derive, competitive schemes could be constructed for a variety of online scheduling problems, including $Rm||C_{max}$ and $Rm||\sum_i C_i^p$ for constant p. Additionally, if we restrict that the processing time of each job is bounded by q, then an optimal online algorithm for $P|p_j \leq q|C_{max}$ could be derived (in $(mq)^{O(mq)}$ time). The readers may refer to Appendix F for details.

5 Concluding Remarks

We provide a new framework for the online over list scheduling problems. We remark that, through such a framework, nearly optimal algorithms could also be derived for other online problems, including the k-server problem (despite that the running time is rather huge, which is exponential).

As nearly optimal algorithms could be derived for various online problems, it becomes a very interesting and challenging problem to consider the hardness of deriving optimal online algorithms. Is there some complexity domain such that finding an optimal online algorithm is *hard* in some sense? For example, given a constant ρ, consider the problem of determining whether there exists an online algorithm for $P||C_{max}$ whose competitive ratio is at most ρ. Could it be answered in time $f(m, \rho)$ for any given function f? We expect the first exciting results along this line, that would open the online area at a new stage.

References

[1] S. Albers. Better bounds for online scheduling. *SIAM Journal on Computing*, 29:459–473, 1999.

[2] S. Albers and M. Hellwig. Semi-online scheduling revisited. *Theoretical Computer Science*, 443:1–9, 2012.

[3] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *Journal of the ACM*, 44(3):486–504, 1997.

[4] A. Avidor, Y. Azar, and J. Sgall. Ancient and new algorithms for load balancing in the l_p norm. *Algorithmica*, 29:422–441, 2001.

[5] B. Awerbuch, Y. Azar, E. Grove, M. Kao, P. Krishnan, and J. Vitter. Load balancing in the l_p norm. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 383–391, 1995.

[6] Y. Azar, J. Naor, and R. Rom. The competitiveness of on-line assignments. In *Proceedings of the 3rdAnnual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 203–210, 1992.

[7] Y. Bartal, A. Fiat, H. Karloff, and R. Vohra. New algorithms for an ancient scheduling problem. *Journal of Computer and System Sciences*, 51:359–366, 1995.

[8] Y. Bartal, H. Karloff, and Rabani Y. A better lower bound for on-line scheduling. *Information Processing Letters*, 50:113–116, 1994.

[9] P. Berman, M. Charikar, and M. Karpinski. On-line load balancing for related machines. *Journal of Algorithms*, 35:108–121, 2000.

[10] B. Chen, A. van Vliet, and G.J. Woeginger. New lower and upper bounds for on-line scheduling. *Operations Research Letters*, 16:221–230, 1994.

[11] T. Cheng, H. Kellerer, and V. Kotov. Semi-on-line multiprocessor scheduling with given total processing time. *Theoretical computer science*, 337(1):134–146, 2005.

[12] Y. Cho and S. Sahni. Bounds for list schedules on uniform processors. *SIAM Journal on Computing*, 9(1):91–103, 1980.

[13] T. Ebenlendr and J. Sgall. A lower bound on deterministic online algorithms for scheduling on related machines without preemption. In *Proceedings of the 9th Workshop on Approximation and Online Algorithms (WAOA)*, pages 102–108, 2012.

[14] U. Faigle, W. Kern, and G. Turán. On the performane of online algorithms for partition problems. *Acta Cybernet*, 9:107–119, 1989.

[15] R. Fleischer and M. Wahl. On-line scheduling revisited. *Journal of Scheduling*, 3:343–353, 2000.

[16] G. Galambos and G. Woeginger. An on-line scheduling heuristic with better worst case ratio than Graham's list scheduling. *SIAM Journal on Computing*, 22:349–355, 1993.

[17] T. Gormley, N. Reingold, E. Torng, and J. Westbrook. Generating adversaries for request-answer games. In *Proceedings of the 11th Annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 564–565, 2000.

[18] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.

[19] E. Günther, O. Maurer, N. Megow, and A. Wiese. A new approach to online scheduling: Approximating the optimal competitive ratio. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2013. To appear.

[20] D. Karger, S.J. Phillips, and E. Torng. A better algorithm for an ancient scheduling problem. In *Proceedings of the 5thAnnual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 132–140, 1994.

[21] J. F. Rudin III. *Improved Bound for the Online Scheduling Problem*. PhD thesis, The University of Texas at Dallas, 2001.

[22] J.F. Rudin III and R. Chandrasekaran. Improved bound for the online scheduling problem. *SIAM Journal on Computing*, 32:717–735, 2003.

[23] J. Sgall. On-line scheduling. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms: The State of the Art*, pages 196–231. Springer, 1998.

A Adding a new job to a scenario

Before we show how a scenario is changed by adding a new job, we first show how a scenario is changed when we scale its jobs using a new factor $T \in SC$ and $T > T_{LB}$.

A.1 Re-computation of a scenario

Let (T_{LB}, ψ) be a real schedule at any time where $\psi = (st_1, st_2, \cdots, st_m)$. If we choose $T > T_{LB}$ to scale jobs, then a big job previously may become a small job (i.e., no greater than $T(1+\epsilon)^{-c_0}$). Suppose $T = T_{LB}(1+\epsilon)^{k\omega}$, then a job with processing time $T_{LB}(1+\epsilon)^j$ is denoted as $T(1+\epsilon)^{j-k\omega}$ now, hence a state $st = (\eta_{-c_0}, \cdots, \eta_\omega)$ of ψ becomes $\hat{st} = (\hat{\eta}_{-c_0}, \cdots, \hat{\eta}_\omega)$ where $\hat{\eta}_i = \eta_{i+k\omega}$ for $i > -c_0$ (we let $\eta_i = 0$ for $i > \omega$), and

$$\hat{\eta}_{c_0} = \frac{\sum_{i=-c_0}^{k\omega-c_0} T_{LB}(1+\epsilon)^i \eta_i}{T(1+\epsilon)^{-c_0}} = \frac{\sum_{i=-c_0}^{k\omega-c_0}(1+\epsilon)^i \eta_i}{(1+\epsilon)^{k\omega-c_0}}.$$

The above computation could be viewed as shifting the state leftwards by $k\omega$ 'bits', and we define a function f_k to represent it such that $f_k(st) = \hat{st}$. Similarly the scenario ψ changes to $\hat{\psi} = (f_k(st_1, \cdots, f_k(st_m))$ and we denote $f_k(\psi) = \hat{\psi}$.

A.2 Adding a new job

Again, let (T_{LB}, ψ) be a real schedule at any time where $\psi = (st_1, st_2, \cdots, st_m)$. Suppose a new job p_n is released and scheduled on machine h where $st_h = (\eta_{-c_0}, \eta_{-c_0+1}, \cdots, \eta_\omega)$, and furthermore, ψ changes to ψ'. We determine the coordinates of ψ' in the following.

Consider p_n. If $p_n \le T_{LB}(1+\epsilon)^\omega$ then we define the addition $st_h + p_n/T_{LB} = \bar{st}_h$ in the following way where $\bar{st}_h = (\bar{\eta}_{-c_0}, \cdots, \bar{\eta}_\omega)$.

- If $p_n/T = (1+\epsilon)^\mu$ for $-c_0+1 \le \mu \le \omega$, then $\bar{\eta}_\mu = \eta_\mu + 1$ and $\bar{\eta}_j = \eta_j$ for $j \ne \mu$.
- If $p_n/T \le (1+\epsilon)^{-c_0}$, then $\bar{\eta}_{-c_0} = \eta_{-c_0} + p_n/(T_{LB}(1+\epsilon)^{-c_0})$ and $\bar{\eta}_j = \eta_j$ for $j \ne -c_0$.

Let $\bar{\psi} = (st_1, \cdots, st_{h-1}, \bar{st}_h, st_{h+1}, \cdots, st_m)$ be a temporal result. If $\bar{\psi}$ is feasible, which implies that $\max\{LD(\bar{\psi})/m, P_{max}(\bar{\psi})\} \in [1, (1+\epsilon)^\omega)$, then $\psi' = \bar{\psi}$. Otherwise $\bar{\psi}$ is infeasible and there are two possibilities.

Case 1. $\max\{1/mLD(\bar{\psi}), P_{max}(\bar{\psi})\} \ge (1+\epsilon)^\omega$. It is not difficult to verify that $\max\{1/mLD(\bar{\psi}), P_{max}(\bar{\psi})\} < (1+\epsilon)^{2\omega}$, thus $f_1(\bar{\psi})$ is feasible and we write $\psi' = f_1(\bar{\psi})$.

Case 2. $1 \le \max\{1/mLD(\bar{\psi}), P_{max}(\bar{\psi})\} < (1+\epsilon)^\omega$ while $LD(\bar{st}_h) > 4(1+\epsilon)^\omega$, i.e., $\bar{st}_h$ is an infeasible state. In this case the competitive ratio of the online algorithm becomes larger than 2. Thus job p_n is never added to st_h if it is scheduled according to an online algorithm with competitive ratio no greater than 2.

Otherwise, $(1+\epsilon)^{k\omega} \le p_n/T_{LB} < (1+\epsilon)^{(k+1)\omega}$ for some $k \ge 1$. It is easy to verify that, by adding p_n to the schedule, the scaling factor becomes $T_{LB}(1+\epsilon)^{k\omega}$. Thus $\psi' = (st'_1, \cdots, st'_m)$ where $st'_j = f_k(st_j)$ for $j \ne h$, and $st'_h = f_k(st_h) + p_n/(T_{LB}(1+\epsilon)^\omega)$.

B Adding a new job to a trimmed-scenario

Notice that a trimmed-scenario could also be viewed as a scenario, thus adding a new job to it could be viewed as adding a new job to a scenario, and then rounding up the coordinates of the resulted scenario to integers. Specifically, we restrict the processing time of the job added is either 0 or $(1+\epsilon)^{\mu}$ for $\mu \geq -c_0$. We will show later that it is possible to put an upper bound on the processing times.

B.1 Re-computation of a trimmed-scenario

To re-compute a trimmed-scenario ϕ, we take ϕ as a scenario with scaling factor $T_{LB} = 1$. Suppose we want to use a new factor $(1+\epsilon)^{\omega}$ to scale jobs, then each trimmed-state of ϕ, say τ, is re-computed as $f_1(\tau)$. Notice that its first coordinate may be fractional, we round it up and let $g_1(\tau) = \lceil f_1(\tau) \rceil$ where $\lceil \vec{v} \rceil$ for a vector means we round each coordinate v_i of $\vec{v}$ to $\lceil v_i \rceil$.

We define g_k iteratively as $g_k(\tau) = g_{k-1}(g_1(\tau))$.

Notice that if τ is feasible (i.e., $LD(\tau) \leq 4(1+\epsilon)^{\omega} + 2(1+\epsilon)^{-c_0}$), then $g_k(\tau)$ is feasible for any $k \geq 1$. Thus, we define $g_k(\phi) = \phi' = (\xi'_1, \xi'_2, \cdots, \xi'_\Lambda)$ where $\xi'_j = \sum_{h:g_k(\tau_h)=\tau_j} \xi_h$. Specifically, if $\{h : g_k(\tau_h) = \tau_j\} = \emptyset$, then $\xi'_j = 0$.

We have the following lemma.

Lemma 5 *For any integer $k \geq 0$, feasible state st_h and feasible trimmed-state τ, the following holds:*

$$(1+\epsilon)^{k\omega} LD(f_k(st_h)) = LD(st_h),$$

$$LD(\tau) \leq (1+\epsilon)^{k\omega} LD(g_k(\tau)) \leq LD(\tau) + \sum_{i=1}^{k} (1+\epsilon)^{-c_0+i\omega} \leq LD(\tau) + 2(1+\epsilon)^{-c_0+k\omega}.$$

The proof is simple through induction.

B.2 Adding a new job

Suppose the feasible trimmed-scenario ϕ becomes ϕ' by adding a new job $p_n = (1+\epsilon)^{\mu}$, and furthermore, the job is added to a machine whose trimmed-state is τ_j. We show how the coordinates of ϕ' is determined.

There are two possibilities.

Case 1. If $-c_0 \leq \mu \leq \omega$, then by adding a new job $p_n = (1+\epsilon)^{\mu}$ to a feasible trimmed-state τ_j, we simply take τ_j as a state and compute $\bar{\tau}_j = \tau_j + p_n$ according to the rule of adding a job to states.

Consider the m trimmed-states of ϕ, we replace τ_j with $\bar{\tau}_j$ while keeping others intact. By doing so a temporal trimmed-scenario $\bar{\phi}$ is generated and we compute $LB(\bar{\phi}) = \max\{1/mLD(\phi) + p_n/m, P_{max}(\phi), p_n\}$. There are three possibilities.

Case 1.1 $LB(\bar{\phi}) < (1+\epsilon)^{\omega} + 2(1+\epsilon)^{-c_0}$ and $LD(\bar{\tau}_j) < 4(1+\epsilon)^{\omega} + 2(1+\epsilon)^{-c_0}$. Then $\bar{\tau}_j$ is a feasible trimmed-state and suppose $\bar{\tau}_j = \tau_{j'}$. Then $\phi' = \bar{\phi}$, i.e., $\phi' = (\xi'_1, \xi'_2, \cdots, \xi'_\Lambda)$ where $\xi'_j = \xi_j - 1$, $\xi'_{j'} = \xi_{j'} + 1$ and $\xi'_l = \xi_l$ for $l \neq j, j'$.

Case 1.2 $LB(\bar{\phi}) < (1+\epsilon)^{\omega}$ and $LD(\bar{\tau}_j) \geq 4(1+\epsilon)^{\omega} + 2(1+\epsilon)^{-c_0}$. Then $\bar{\tau}_j$ is infeasible and $\phi' = \phi_{\Gamma+1}$.

Case 1.3 $LB(\bar{\phi}) \geq (1+\epsilon)^{\omega}$. It can be easily verified that $LB(\bar{\phi}) < (1+\epsilon)^{2\omega}$. Notice that $g_1(\bar{\tau}_j)$ is always feasible, thus $\phi' = g_1(\bar{\phi})$, i.e., for each trimmed-state τ of $\bar{\phi}$, we compute $g_1(\tau)$. Since $g_1(\tau)$ is always feasible, they made up of a feasible trimmed-scenario ϕ'.

Remark. There might be intersection between Case 1 and Case 3. Indeed, if $(1+\epsilon)^{\omega} \leq LB(\bar{\phi}) < (1+\epsilon)^{\omega} + 2(1+\epsilon)^{-c_0}$, and $\bar{\tau}$ is feasible, then by adding p_n the trimmed-scenario ϕ changes into $\bar{\phi} = \phi'$ according to Case 1 and $g_1(\phi')$ according to Case 3. Here both ϕ' and $g_1(\phi')$ are feasible trimmed-scenarios.

This is the only case that $\phi + p_n$ may yield two different solutions. In the next section we will remove ϕ if both ϕ and $g_1(\phi)$ are feasible. By doing so $\phi + p_n$ yields a unique solution, but currently we just keep both of them so that Theorem 2 could be proved.

Case 2. If $(1+\epsilon)^{k\omega} \leq \mu < (1+\epsilon)^{(k+1)\omega}$ then again we take τ_j as a state and compute $\bar{\tau}_j = g_k(\tau_j) + p_n/(1+\epsilon)^{k\omega}$.

We re-compute ϕ as $g_k(\phi) = (\hat{\xi}_1, \hat{\xi}_2, \cdots, \hat{\xi}_\Lambda)$. Then we replace one trimmed-state $g_k(\tau_j)$ with $\bar{\tau}_j$ and this generates ϕ'. It is easy to verify that ϕ' is feasible.

Remark 2. Notice that the number of possible processing times of job p_n could be infinite, however, we show that it is possible to further restrict it to be some constant.

Let $p_n = (1+\epsilon)^{\mu}$. Let μ_0 be the smallest integer such that $(1+\epsilon)^{\mu_0} \geq 4(1+\epsilon)^{\omega+c_0+1}$. If $\mu = k\omega + l$ with $k \geq \lceil \mu_0/\omega \rceil$ and $0 \leq l \leq \omega - 1$, then ϕ is re-computed as $g_k(\phi)$. Notice that for any feasible trimmed-state τ, $LD(\tau) \leq 4(1+\epsilon)^{\omega} + 2(1+\epsilon)^{-c_0} < 4(1+\epsilon)^{\omega+1}$, thus $LD(g_k(\tau)) \leq (1+\epsilon)^{-c_0}$, which implies that $g_k(\tau) = (0,0,\cdots,0)$ if $\tau = (0,0,\cdots,0)$ and $g_k(\tau) = (1,0,0,\cdots,0)$ otherwise. Thus, $g_k(\phi) = g_{\lceil \mu_0/\omega \rceil}(\phi)$.

The above analysis shows that by adding a job with processing time $p_n = (1+\epsilon)^{k\omega+l}$ for $k \geq \lceil \mu_0/\omega \rceil$ and $0 \leq l \leq \omega - 1$ to any feasible trimmed-scenario ϕ is equivalent to adding a job with processing time $p_n = (1+\epsilon)^{\lceil \mu_0/\omega \rceil \omega + l}$ to ϕ.

Thus, when adding a job to a trimmed-scenario, we may restrict that $p_n \in R = \{0, (1+\epsilon)^{-c_0}, \cdots, (1+\epsilon)^{\lceil \mu_0/\omega \rceil + \omega - 1}\}$.

C Simulating transformations between scenarios

The whole section is devoted to prove the following theorem.

Theorem 2 *Let ϕ be the simulating-scenario of a feasible scenario ψ. If according to some online algorithm (T, ψ) changes to $(T', \bar{\psi})$ by adding a job $p_n \neq 0$, then ϕ could be transformed to $\bar{\phi}$ ($\bar{\phi} \neq \phi_0, \phi_\Gamma$) by adding a job $p'_n \in R = \{0, (1+\epsilon)^{-c_0}, \cdots, (1+\epsilon)^{\lceil \mu_0/\omega \rceil + \omega - 1}\}$ such that $\bar{\phi}$ is a simulating-scenario of $\bar{\psi}$.*

Let $\tau_{\theta(h)}$ in ϕ be the simulating-state of st_h in ψ. Before we give the proof, we first present a lemma that would be used later.

Lemma 6 *Let ϕ be a simulating-scenario of ψ. For any $k \geq 1$, if $f_k(st_h) = (\eta'_{-c_0}, \eta'_{-c_0+1}, \cdots, \eta'_{\omega})$ and $g_k(\tau_{\theta(h)}) = (\nu'_{-c_0}, \nu'_{-c_0+1}, \cdots, \nu'_{\omega})$, then $\nu'_i = \eta'_i$ for $i > -c_0$ and $\eta'_{-c_0} \leq \nu'_{-c_0} \leq \eta'_{-c_0} + 2$.*

Proof. Let $st_h = (\eta_{-c_0}, \eta_{-c_0+1}, \cdots, \eta_{\omega})$ and $\tau_{\theta(h)} = (\nu_{-c_0}, \nu_{-c_0+1}, \cdots, \nu_{\omega})$. We first prove the lemma for $k = 1$.

It is easy to verify that $\nu'_i = \eta'_i$ for $i > -c_0$. Furthermore,

$$\begin{aligned}
\nu'_{-c_0} &= \lceil \frac{\sum_{i=-c_0}^{\omega-c_0}(1+\epsilon)^i \nu_i}{(1+\epsilon)^{\omega-c_0}} \rceil \\
&\le \frac{\sum_{i=-c_0+1}^{\omega-c_0}(1+\epsilon)^i \eta_i + (1+\epsilon)^{-c_0}(\eta_{-c_0}+2)}{(1+\epsilon)^{\omega-c_0}} + 1 \\
&\le \eta'_{-c_0} + 1 + 2(1+\epsilon)^{-\omega} < \eta'_{-c_0} + 2
\end{aligned}$$

Thus the lemma holds for $k = 1$.

If the lemma holds for $k = k_0$, then it also holds for $k = k_0 + 1$. The proof is the same. □

Now we come to the proof of Theorem 2.

Proof. Let $\psi = (st_1, st_2, \cdots, st_m)$ and $\phi = (\xi_1, \xi_2, \cdots, \xi_\Lambda)$. Recall that $\tau_{\theta(i)}$ is the simulating-state of st_i in ϕ.

Notice that $LD(st_i) \le LD(\tau_{\theta(i)}) \le LD(st_i) + 2(1+\epsilon)^{-c_0}$, it follows that $1/mLD(\psi) \le 1/mLD(\phi) \le 1/mLD(\psi) + 2(1+\epsilon)^{-c_0}$. Meanwhile, $P_{max}(\psi) = P_{max}(\phi)$ as long as $\psi \ne (0, 0, \cdots, 0)$.

Suppose job n is assigned to machine h in the real schedule. Let $st_h = (\eta_{-c_0}, \cdots, \eta_\omega)$ and $\tau_{\theta(h)} = (\nu_{-c_0}, \cdots, \nu_\omega)$. Recall that $\eta_{-c_0} \le \nu_{c_0} \le \eta_{-c_0} + 2$ and $\eta_i = \nu_i$ for $i > c_0$.

There are two possibilities.

Case 1.

$$p_n/T \le (1+\epsilon)^\omega.$$

Let $st'_h = st_h + p_n/T = (\eta'_{-c_0}, \cdots, \eta'_\omega)$. We define p'_n in the following way.

- If $p_n/T = (1+\epsilon)^\mu$ for $-c_0 + 1 \le \mu \le \omega$, then $p'_n = p_n/T$.
- If $p_n/T \le (1+\epsilon)^{-c_0}$,
  - $\eta'_{-c_0} \le \nu_{-c_0}$, then $p'_n = 0$.
  - $\eta'_{-c_0} > \nu_{-c_0}$, then $p'_n = (1+\epsilon)^{-c_0}$.

Let $\tau'_{\theta(h)} + p'_n = (\nu'_{-c_0}, \cdots, \nu'_\omega)$, then $\nu'_{-c_0} = \nu_{-c_0} + p'_n/((1+\epsilon)^{-c_0})$, in both cases $\eta'_{-c_0} \le \nu'_{-c_0} \le \eta'_{-c_0} + 2$.

By adding p_n to ψ, the scaling factor may or may not be changed.

If $T = T'$, the state of machine h in $\bar{\psi}$ is st'_h. We consider $\tau_{\zeta(h)} + p'_n$. Since $LD(\tau_{\zeta(h)} + p'_n) - LD(st'_h) \le LD(\tau_{\zeta(h)}) - LD(st_h) \le 2(1+\epsilon)^{-c_0}$, and st'_h is a feasible state, $\tau_{\zeta(h)} + p'_n$ is also a feasible trimmed state. Meanwhile $\max\{1/mLD(\psi'), P_{max}(\psi')\} < (1+\epsilon)^\omega$, thus by adding p'_n to ϕ, the scaling factor of the trimmed-scenario is also not updated, which implies that the trimmed-state of machine h in $\bar{\phi}$ is $\tau_{\zeta(h)} + p'_n$. It can be easily verified that in this case, $\bar{\phi}$ is the simulating-scenario of $\bar{\psi}$.

Otherwise $T' > T$ and the state of machine h is $f_1(st'_h)$ in $\bar{\psi}$. We compute $LB' = \max\{1/mLD(\phi) + p'_n/m, P_{max}(\phi), p'_n\}$. Since $LB = \max\{1/mLD(\psi) + p_n/m, P_{max}(\psi), p_n\} > (1+\omega)^\omega$, it follows directly that $LB' > (1+\omega)^\omega$. Meanwhile $LB' < (1+\omega)^{2\omega}$, thus the trimmed-state of machine h in $\bar{\phi}$ is $g_1(\tau'_{\zeta(h)} + p'_n)$.

We compare st'_h and $\tau'_{\zeta(h)}+p'_n=(\nu'_{-c_0},\cdots,\nu'_\omega)$. Obviously $\eta'_l=\nu'_l$ for $l\neq -c_0$ and $\eta_{c'_0}\le\nu'_{-c_0}\le\eta'_{-c_0}+2$. According to Lemma 6, $g_1(\tau'_{\zeta(h)}+p'_n)$ is a simulating-state of $f_1(st'_h)$, which implies that $\bar{\phi}$ is a simulating-scenario of $\bar{\psi}$.

Remark. Recall that when $(1+\epsilon)^\omega\le LB'<(1+\epsilon)^\omega+2(1+\epsilon)^{-c_0}$, $\phi+p'_n$ may yield two solutions $\hat{\phi}$ and $g_1(\hat{\phi})$, as we have claimed. Our above discussion chooses $\hat{\phi}$ if the scaling factor of the real schedule does not change, and chooses $g_1(\hat{\phi})$ when the the scaling factor of the real schedule changes.

Case 2. For some $k\ge 1$,

$$(1+\epsilon)^{k\omega}\le p_n/T<(1+\epsilon)^{(k+1)\omega}.$$

Then we define $p'_n=p_n/T$ at first.

Let $f_k(st_h)=(\eta'_{-c_0},\cdots,\eta'_\omega)$, $g_k(\tau_{\zeta(h)})=(\nu'_{-c_0},\cdots,\nu'_\omega)$, then according to Lemma 6 we have $\eta'_i=\nu'_i$ for $-c_0<i\le\omega$ and $\eta'_{-c_0}\le\nu'_{-c_0}\le\eta'_{-c_0}+2$. Then it follows directly that $g_k(\tau_{\zeta(h)})+p'_n$ is a simulating-state of $f_k(st_h)+p_n$. Thus, by adding p'_n, $\bar{\phi}$ is a simulating-scenario of $\bar{\psi}$.

Furthermore, if $p'_n>(1+\epsilon)^{\lceil\mu_0/\omega\rceil+\omega-1}$, then suppose $p'_n=(1+\epsilon)^{k'\omega+l}$ for some $k'\ge\lceil\mu_0/\omega\rceil$ and $0\le l\le\omega-1$. Due to our previous analysis, p'_n could be replaced by a job with processing time $p''_n=(1+\epsilon)^{\lceil\mu_0/\omega\rceil+l}$. The trimmed-scenario ϕ still transforms into $\bar{\phi}$ by adding p''_n. □

D Deletion of equivalent trimmed-scenarios

Recall that the addition $\phi+p_n$ may yield two solutions, ϕ' and $g_1(\phi')$ where both of them are feasible. To make the result unique, ϕ' is deleted from Φ if $g_1(\phi')$ is feasible and Φ' is the set of the remaining trimmed-scenarios.

We have the following simple lemma.

Lemma 7 *If ϕ and $g_1(\phi)$ are both feasible trimmed-scenarios, then $|\rho(\phi)-\rho(g_1(\phi))|\le O(\epsilon)$.*

With fewer trimmed-scenarios, Theorem 2 may not hold, however, we have the following lemma.

Lemma 8 *Suppose by releasing job n with $p_n\in R$ and scheduling it onto a certain machine, the feasible trimmed-scenario ϕ changes to $\hat{\phi}$. Furthermore, $g_1(\phi)$ is also feasible. Then there exists $p'_n\in R$ such that by scheduling it on the same machine, $g_1(\phi)$ changes to $\bar{\phi}$ and furthermore, either $\bar{\phi}=\hat{\phi}$ or $\bar{\phi}=g_1(\hat{\phi})$.*

Proof. Suppose job n is scheduled onto a machine of trimmed-state $\tau=(\nu_{-c_0},\cdots,\nu_\omega)$ in ϕ, then we put p'_n onto a machine of trimmed-state $g_1(\tau)=(\nu'_{-c_0},\cdots,\nu'_\omega)$ in $g_1(\phi)$. If $p_n=0$ then obviously we can choose $p'_n=0$. Otherwise let $p_n=(1+\epsilon)^\mu$ and there are three possibilities.

Case 1. $\mu\le\omega-c_0$.

If by adding p_n, the scaling factor of ϕ does not change, then we compare $\nu'_{-c_0}=\lceil\frac{\sum_{i=-c_0}^{k\omega-c_0}(1+\epsilon)^i\nu_i}{(1+\epsilon)^{\omega-c_0}}\rceil$ with $y=\lceil\frac{\sum_{i=-c_0}^{k\omega-c_0}(1+\epsilon)^i\nu_i+(1+\epsilon)^\mu}{(1+\epsilon)^{\omega-c_0}}\rceil\le\nu'_{-c_0}+1$. If $\nu'_{-c_0}=y$, then $p'_n=0$. Otherwise $y=\nu'_{-c_0}+1$, then $p'_n=(1+\epsilon)^{-c_0}$. It can be easily verified that $g_1(\tau)+p'_n=g_1(\tau+p_n)$ and $g_1(\hat{\phi})=g_1(\phi)+p'_n$.

Otherwise by adding p_n the scaling factor of ϕ increases, then we define p'_n in the same way and it can be easily verified that $\hat{\phi}=g_1(\phi)+p'_n$.

Case 2. $\omega-c_0<\mu\le 2\omega$.

In this case we define $p'_n = (1+\omega)^{\mu-\omega}$ and the proof is similar to the previous case.

Notice that in both case 1 and case 2, $p'_n \leq (1+\omega)^{\omega}$. As $LD(g_1(\tau)) \leq 4 + 2(1+\epsilon)^{-c_0-\omega}$, $LD(g_1(\tau)) + p'_n \leq 4(1+\epsilon)^{\omega}$, thus we can add p'_n to $g_1(\tau)$ directly (without changing the scaling factor). Furthermore, $\max\{1/m[LD(g_1(\phi)) + p'_n], P_{max}(g_1(g_1(\phi))), p'_n\} \leq (1+\epsilon)^{\omega}$, thus by adding p'_n to $g_1(\phi)$, the scaling factor does not change, thus in both cases, $\bar{\phi} = g_1(\phi) + p'_n$.

Case 3. $\mu > 2\omega$.

Suppose $\mu = k\omega + l$ with $k \geq 2$ and $0 \leq l \leq \omega - 1$. Then $p'_n = (1+\epsilon)^{\mu-\omega}$. According to the definition of g_k, $g_k(\phi) = g_{k-1}(g_1(\phi))$, thus $\bar{\phi} = \hat{\phi}$.

□

Combining Theorem 2 and Lemma 8, we have the following theorem.

Theorem 3 *Let $\phi \in \Phi'$ be the simulating-scenario or shifted simulating-scenario of a feasible scenario ψ. If according to some online algorithm (T, ψ) changes to $(T', \bar{\psi})$ by adding a job $p_n \neq 0$, then ϕ could be transformed to $\bar{\phi} \in \Phi'$ ($\bar{\phi} \neq \phi_0, \phi_\Gamma$) by adding a job $p'_n \in R = \{0, (1+\epsilon)^{-c_0}, \cdots, (1+\epsilon)^{\lceil \mu_0/\omega \rceil + \omega - 1}\}$ such that $\bar{\phi}$ is a simulating-scenario or shifted simulating-scenario of $\bar{\psi}$.*

E The nearly optimal strategies for the adversary and the scheduler

E.1 The nearly optimal strategy for the adversary

We prove in this subsection that, by releasing at most n_0 jobs, the adversary can ensure that there is no online algorithm whose competitive ratio is less than $\rho^* - O(\epsilon)$.

We play the part of the adversary.

Consider G_{n_0}. Notice that $\rho^* = \rho_{n_0}(s_0^0) = \max_j\{\rho_{n_0}(a_{0,j}^0) : a_{0,j}^0 \in N(s_0^0)\}$, thus there exists some j_0 such that $a_{0,j_0}^0 \in N(s_0^0)$ and $\rho_{n_0}(a_{0,j_0}^0) = \rho^*$.

We release a job with processing time α_{j_0}. Suppose due to any online algorithm whose competitive ratio is no greater than 2, this job is scheduled onto a certain machine so that the scenario becomes ψ, then according to Theorem 2 and the construction of the graph, there exists some s_k^1 incident to a_{0,j_0}^0 such that either ϕ_k is a simulating-scenario of ψ, or ϕ_k is a shifted simulating-scenario of ψ. As $\rho_{n_0}(a_{0,j_0}^0) = \min_k\{\rho_{n_0}(s_k^1) : s_k^1 \in N(a_{0,j_0}^0)\}$, it follows directly that $\rho_{n_0}(s_k^1) \geq \rho^*$. If $\rho(\phi_k) = \rho_{n_0}(s_k^1) \geq \rho^*$, then we stop and it can be easily seen that the instant approximation ratio of ψ is at least $\rho^* - O(\epsilon)$ (by Lemma 1). Otherwise we go on to release jobs.

Suppose after releasing $h-1$ jobs the current scenario is ψ and ϕ_i is its simulating-scenario or shifted simulating-scenario, furthermore, $\rho_{n_0}(s_i^{h-1}) \geq \rho^*$. As $\rho^* \leq \rho_{n_0}(s_i^{h-1}) = \max_j\{\rho_{n_0}(a_{ij}^{h-1}) : a_{ij}^0 \in N(s_i^{h-1})\}$, thus there exists some j_0 such that $a_{ij_0}^{h-1} \in N(s_i^{h-1})$ and $\rho_{n_0}(a_{ij_0}^0) \geq \rho^*$.

We release the h-th job with processing time α_{j_0}. Again suppose this job is scheduled onto a certain machine so that the scenario becomes ψ', then there exists some s_k^h incident to $a_{ij_0}^{h-1}$ such that ϕ_k is either a simulating-scenario or a shifted simulating-scenario of ψ'. As $\rho_{n_0}(a_{ij_0}^{h-1}) = \min_k\{\rho_{n_0}(s_k^h) : s_k^h \in N(a_{ij_0}^0)\}$, it follows directly that $\rho_{n_0}(s_k^h) \geq \rho^*$. If $\rho(\phi_k) = \rho_{n_0}(s_k^h) \geq \rho^*$, then we stop and it can be easily seen that the instant approximation ratio of ψ' is at least $\rho^* - O(\epsilon)$. Otherwise we go on to release jobs.

Since $\rho(\phi_k) = \rho_{n_0}(s_k^{n_0})$, we stop after releasing at most n_0 jobs.

E.2 The nearly optimal online algorithm

We play the part of the scheduler.

Notice that

$$\rho_{n_0+1}(a_{ij}^0) = \min_k\{\rho_{n_0+1}(s_k^1) : s_k^1 \in N(a_{ij}^0)\} = \min_k\{\rho(s_k^1) : s_k^1 \in N(a_{ij}^0)\},$$

$$\rho(s_i^0) = \rho_{n_0+1}(s_i^0) = \max_j\{\rho_{n_0+1}(a_{ij}^0) : a_{ij}^0 \in N(s_i^0)\}.$$

Suppose the current scenario is ψ with scaling factor T. Let $\phi_i \in \Phi'$ be its simulating-scenario or shifted simulating-scenario, and furthermore, $\rho(s_i^0) \le \rho^*$.

Let p_n be the next job the adversary releases. We apply lazy scheduling first, i.e., if by scheduling p_n onto any machine, ψ changes to ψ' (the scaling factor does not change) while ϕ_i is still a simulating-scenario or shifted simulating-scenario of ψ', we always schedule p_n onto this machine.

Otherwise, According to Theorem 2 and Lemma 8, $p'_n(h)$ could be constructed such that if ψ changes to ψ' by adding p_n to machine h, then ϕ changes to ϕ' by adding p'_n to the same machine such that ϕ' is a simulating-scenario or shifted simulating-scenario of ψ'. Notice that the processing time of $p'_n(h)$ may also depend on the machine h.

We show that, if p_n could not be scheduled due to lazy scheduling, then $p'_n(h) = p'_n$ for every h. To see why, we check the proofs of Theorem 2 and Lemma 8. We observe that, if $p'_n(h) \ge (1+\epsilon)^{-c_0+1}$ for some h, then $p'_n(h) = p'_n$ for every h (the processing time $p'_n(h)$ only depends on p_n/T). Otherwise, it might be possible that $p'_n(h_1) = 0$ for some h_1 while $p'_n(h_2) = (1+\epsilon)^{-c_0}$ for another h_2. However, if this is the case then p_n should be scheduled on machine h_1 according to lazy scheduling, which is a contradiction. Thus, $p'_n(h) = (1+\epsilon)^{-c_0}$ for every h.

Now we decide according to G_{n_0+1} which machine p_n should be put onto.

As $p'_n \in R$, let $\alpha_{j_0} = p'_n$, then we consider $\rho_{n_0+1}(a_{ij_0}^0) = \min_k\{\rho_{n_0+1}(s_k^1) : s_k^1 \in N(a_{ij_0}^0)\}$. Recall that $\rho(s_i^0) \le \rho^*$ according to the hypothesis, then $\rho_{n_0+1}(a_{ij_0}^0) \le \rho^*$, which implies that there exists some $s_{k_0}^1$ incident to a_{ij_0} such that $\rho_{n_0+1}(s_{k_0}^1) = \rho(s_{k_0}^0) \ge \rho^*$. Thus, we can schedule p'_n to a certain machine, say, machine h_0, so that ϕ_i transforms to ϕ_{k_0}. And thus in the real schedule we schedule p_n onto machine h_0. Let ψ' be the current scenario, then ϕ_{k_0} is its simulating-scenario or shifted simulating-scenario with $\rho(s_{k_0}^0) \le \rho^*$.

Thus, we can always carry on the above procedure. Since the instant approximation ratio of each simulating-scenario or shifted simulating-scenario is no greater than ρ^*, the instant approximation ratio of the corresponding scenario is also no greater than $\rho^* + O(\epsilon)$.

F Extensions

We show in this section that our method could be extended to provide approximation schemes for various problems. Specifically, we consider $Rm||C_{max}$, $Rm||\sum_h C_h^p$ for some constant $p \ge 1$ (and as a consequence $Qm||C_{max}$, $Qm||\sum_h C_h^p$ and $Pm||\sum_h C_h^p$ could also be solved). We mention that, if we restrict that the number of machines m is a constant (as in the case $Rm||C_{max}$ and $Rm||\sum_h C_h^p$), then our method could be simplified.

We also consider the semi-online model $P|p_j \le q|C_{max}$ where the processing time of each job released is at most q. In this case an optimal algorithm could be derived in $(mq)^{O(mq)}$ time. Notice that our previous discussions focus on finding nearly optimal online algorithms, however, for

online problems, we do not know much about optimal algorithms. Only the special cases $P2||C_{max}$ and $P3||C_{max}$ are known to admit optimal algorithms. Unlike the corresponding offline problems which always admit exact algorithms (sometimes with exponential running times), we do not know whether there exists such an algorithm for online problems. Consider the following problem, does there exist an algorithm which determines whether there exists an online algorithm for $P||C_{max}$ whose competitive ratio is no greater than ρ. We do not know which complexity class this problem belongs to. An exact algorithm, even with running time exponential in the input size, would be of great interest.

Related work. For the objective of minimizing the makespan on related and unrelated machines, the best known results are in table 1. There is a huge gap between the upper bound and lower bound except for the special case $Q_2||C_{max}$. However, the standard technique for $Q_2||C_{max}$ becomes extremely complicated and can hardly be extended for 3 or more machines.

For the objective of $(\sum_h C_h^p)^{1/p}$, i.e., the L_p norm, not much is known. See table 1 for an overview. We further mention that when $p = 2$, List Scheduling is of competitive ratio $\sqrt{4/3}$ [4].

Table 1: Lower and upper bounds on the competitive ratio for deterministic

problems	lower bounds	upper bounds
$Q\|\|C_{\max}$	2.564 [13]	5.828 [9]
$Q2\|\|C_{\max}$	$(2s+1)/(s+1)$ for $s \le 1.61803$	$(2s+1)/(s+1)$ for $s \le 1.61803$,
	$1+1/s$ for $s \ge 1.61803$ [12]	$1+1/s$ for $s \ge 1.61803$ [12]
$R\|\|C_{\max}$	$\Omega(\log m)$ [6]	$O(\log m)$ [3]
$P\|\|(\sum_h C_h^p)^{1/p}$		$2-\Theta(\ln p/p)$ [4]
$R\|\|(\sum_h C_h^p)^{1/p}$		$O(p)$ [5]

Much of the previous work is directed for semi-online models of scheduling problems where part of the future information is known beforehand, and most of them assume that the total processing time of jobs (instead of the largest job) is known. For such a model, the best known upper bound is 1.6 [11] and the best known lower bound is 1.585 [2].

F.1 $Rm||C_{max}$

In this case, we can restrict beforehand that the processing time of each job, say, j, on machine h ($1 \le h \le m$) is $p_{jh} \in \{(1+\epsilon)^k : k \ge 0, k \in \mathbb{N}\}$. There is a naive algorithm Al_0 that puts every job on the machine with the least processing time, and it can be easily seen that the competitive ratio of this algorithm is m. Since m is a constant, it is a constant competitive ratio online algorithm, and thus we may restrict on the algorithms whose competitive ratio is no greater than m.

Given any real schedule, we may first compute the makespan of the schedule by applying Al_0 on the instance and let it be $Al_0(C_{max})$, then we define $LB = Al_0(C_{max})/m$ and find a scaling factor $T \in SC$ such that $T \le LB < T(1+\epsilon)^\omega$. Similarly as we do in the previous sections, we can then define a state for each machine of the real schedule with respect to T and then a scenario by combining the m states. Since $OPT \le mT(1+\epsilon)^\omega$, if the real schedule is produced by an online algorithm whose competitive ratio is no greater than m, then the load of each machine is bounded by $m^2T(1+\epsilon)^\omega$, and this allows us to bound the number of different feasible states by

some constant, and the number of all different feasible scenarios is also bounded by a constant (depending on m and $1/\epsilon$).

We can then define trimmed-states and trimmed-scenarios in the same way as before. Specifically, a trimmed-state is combined of m trimmed-states directly (it is much simpler since the number of machines is a constant). Again, a feasible trimmed-state is a trimmed-state whose load could be slightly larger than $m^2T(1+\epsilon)^\omega$ (to include two additional small jobs), and a feasible trimmed-scenario is a trimmed-scenario such that every trimmed-state is feasible.

Transformations between scenarios and trimmed-scenarios are exactly the same as before and we can also construct a graph to characterize the transformations between trimmed-scenarios, and use it to approximately characterize the transformation between scenarios. All the subsequent arguments are the same.

F.2 $Rm||\sum_h C_h^p$ when $p \geq 1$ is a constant

Here C_h denotes the load of machine h.

Again we can restrict beforehand that the processing time of each job, say, j, on machine h ($1 \leq h \leq m$) is $p_{jh} \in \{(1+\epsilon)^k : k \geq 0, k \in \mathbb{N}\}$. Consider the naive algorithm Al_0 that puts every job on the machine with the least processing time and let $C_h(Al_0)$ be the load of machine h due to this algorithm. Since x^p is a convex function, we know directly that $OPT \geq m(\frac{\sum_{h=1}^m C_h(Al_0)}{m})^p \geq \frac{\sum_{h=1}^m C_h(Al_0)^p}{m}$ and thus the competitive ratio of Al_0 is also m and again we may restrict on the algorithms whose competitive ratio is no greater than m.

Given any real schedule, we may first compute the objective function of the schedule by applying Al_0 on the instance and let it be $Al_0(\sum_h C_h^p)$, then we define $LB = [Al_0(\sum_h C_h^p)/m]^{1/p}$ and find a scaling factor $T \in SC$ such that $T \leq LB < T(1+\epsilon)^\omega$. Consider any schedule produced by an online algorithm whose competitive ratio is no greater than m, then its objective value should be bounded by $mAl_0(\sum_h C_h^p)$, which implies that the load of each machine in this schedule is bounded by $[mAl_0(\sum_h C_h^p)]^{1/p} = m^{2/p}LB$. Again using the fact that m is a constant, we can then define a state for each machine of the real schedule with respect to T and then a scenario by combining the m states. Trimmed-states and trimmed-scenarios are defined similarly, all the subsequent arguments are the same as the previous subsection.

Remark. Our method, however, could not be extended in a direct way to solve the more general model $Rm||\sum_h f(C_h)$ if the function f fails to satisfy the property that $f(ka)/f(kb) = f(a)/f(b)$ for any $k > 0$. This is because we neglect the scaling factor when we construct the graph G and compute the instant approximation ratio for each trimmed-scenario. Indeed, the instant approximation ratio is not dependent on the scaling factor for all the objective functions (i.e., C_{max} and $\sum_h C_h^p$) we consider before, however, if such a property is not satisfied, then the instant approximation ratio depends on the scaling factor and our method fails.

F.3 $P|p_j \leq q|C_{max}$

We show in this subsection that, the semi-online scheduling problem $P|p_j \leq q|C_{max}$ in which the largest job is bounded by some integer ζ (the value q is known beforehand), admits an exact online algorithm.

Again we use the previous framework to solve this problem. The key observation is that, in

such a semi-online model, we can restrict our attentions only on bounded instances in which the total processing time of all the jobs released by the adversary is bounded by $2m\zeta$. It is easy to verify that, if we only consider bounded instances, then we can always use a ζ-tuple to represent the jobs scheduled on each machine. This is the state for a machine and there are at most $(2mq)^q$ different states. Combining the m states generates scenarios, and there are at most $(2mq)^{mq}$ different scenarios, and thus we can construct a graph to represent the transformations between these scenarios and find the optimal online algorithm using the same arguments.

We prove the above observation in the following part of this subsection.

We restrict that $q \geq 2$ since we assume that the processing time of each job is some integer, and $q = 1$ would implies that the adversary only releases jobs of processing time 1, and list scheduling is the optimal algorithm.

When $q \geq 2$, we know that the competitive ratio of any online algorithm is no less than 1.5. To see why, suppose there are only two machines and the adversary releases at first two jobs, both of processing time 1. Any online algorithm that puts the two jobs on the same machine would have a competitive ratio at least 2. Otherwise suppose an online algorithm puts the two jobs on separate machines, then the adversary releases a job of processing time 2, and it can be easily seen that the competitive ratio of this online algorithm is at least 1.5.

We use I to denote a list of jobs released by the adversary (one by one due to the sequence), and this is an instance. We use $LD(I)$ to denote the total processing time of jobs in I. Let Ω be the set of all instances and $\Omega_B = \{I | LD(I)/m \leq 2p\}$ be the set of bounded instances. Let A be the set of all the online algorithms. Let $Al \in A$ be any online algorithm, it can be easily seen that its competitive ratio ρ_{Al} is defined as

$$\rho_{Al} = \sup_{I \in \Omega} \frac{Al(I)}{OPT(I)},$$

where $OPT(I)$ is the makespan of the optimal (offline) solution for the instance I and $Al(I)$ is the makespan of the solution produced by the algorithm.

The goal of this subsection is to find an algorithm Al^* such that

$$\rho_{Al^*} = \inf_{Al \in A} \sup_{I \in \Omega} \frac{Al(I)}{OPT(I)}.$$

On the other hand, according to our previous discussion, we can find an algorithm Al_B^* such that

$$\rho_{Al_B^*} = \inf_{Al \in A} \sup_{I \in \Omega_B} \frac{Al(I)}{OPT(I)}.$$

Notice that when we restrict our attentions on bounded instances, the algorithm we find may be only defined for $I \in \Omega_B$, we extend it to solve all the instances in the following way. We use LS to denote the list scheduling. Given any algorithm Al which can produce a solution for any instance $I \in \Omega_B$, we use $Al \circ LS$ to denote the LS-composition of this algorithm where the algorithm $Al \circ LS$ operates in the following way.

Recall that $I \in \Omega$ is a list of jobs and let it be $(p_1, p_2, \cdots, p_n)$ where $p_j \geq 1$. If $I \in \Omega_B$, then $Al \circ LS$ schedules jobs in the same way as Al. Otherwise let j_0 be the largest index such that $\sum_{j=1}^{j_0} p_j \leq 2m\zeta$, $Al \circ LS$ schedules job 1 to job j_0 in the same way as Al, and schedules the

subsequent jobs according to list scheduling, i.e., when p_j $(j > j_0)$ is released, we put this job onto the machine with the least load currently.

Thus, the algorithm $Al \circ LS$ could be viewed as a combination of Al and LS, and we only require that Al is defined for instances of Ω_B.

Lemma 9 *For any* $Al \in A$,

$$\rho_{Al\circ LS} \le \sup_{I\in\Omega_B} \frac{Al(I)}{OPT(I)}.$$

Proof. Consider $I = (p_1, p_2, \cdots, p_n) \notin \Omega_B$ and suppose j_0 is the largest index such that $\sum_{j=1}^{j_0} p_j \le 2mp$. Let $I_B = (p_1, p_2, \cdots, p_{j_0})$, then obviously $OPT(I) \ge OPT(I_B)$.

Consider $Al\circ LS(I)$. If $Al\circ LS(I) = Al(I_B)$, then obviously $Al\circ LS(I)/OPT(I) \le Al(I_B)/OPT(I_B)$.

Otherwise $Al \circ LS(I) > Al(I_B)$, and let $h > j_0$ be the job whose completion time achieves $Al \circ LS(I)$. Since h is scheduled due to the LS-rule, we know that $Al \circ LS(I) \le LD(I)/m + p_h$. Notice that $OPT \ge LD(I)/m \ge 2p$, thus $Al \circ LS(I)/OPT(I) \le 1.5$. Thus we have

$$\rho_{Al\circ LS} \le \max\{\sup_{I_B\in\Omega_B} Al(I_B)/OPT(I_B), 1.5\}.$$

Recall that we have shown in the previous discussion that $\sup_{I_B\in\Omega_B} Al(I_B)/OPT(I_B) \ge 1.5$, thus $\rho_{Al\circ LS} \le \sup_{I\in\Omega_B} \frac{Al(I)}{OPT(I)}$. □

The above lemma shows that $\rho_{Al^*_B\circ LS} \le \rho_{Al^*_B}$. Meanwhile it is easy to see that $\rho_{Al^*_B\circ LS} \ge \rho_{Al^*_B}$, thus $\rho_{Al^*_B\circ LS} = \rho_{Al^*_B}$.

We prove in the following part that $\rho_{Al^*_B} = \rho_{Al^*}$, and thus $Al^*_B \circ LS$ is the best algorithm for the semi-online problem.

Obviously $\sup_{I\in\Omega} Al(I)/OPT(I) \ge \sup_{I\in\Omega_B} Al(I)/OPT(I)$, thus $\rho_{Al^*} \ge \rho_{Al^*_B}$.

On the other hand, let $A \circ LS = \{Al \circ LS : Al \in A\} \subset A$,

$$\inf_{Al\in A} \rho_{Al} \le \inf_{Al\in A\circ LS} \rho_{Al\circ LS}.$$

According to Lemma 9, for any $I \in \Omega$,

$$\inf_{Al\in A\circ LS} \rho_{Al\circ LS} \le \inf_{Al\in A\circ LS} \sup_{I\in\Omega_B} \frac{Al(I)}{OPT(I)} = \inf_{Al\in A} \sup_{I\in\Omega_B} \frac{Al(I)}{OPT(I)},$$

thus $\rho_{Al^*} \le \rho_{Al^*_B}$, which implies that $\rho_{Al^*} = \rho_{Al^*_B}$.